



Software Engineering Institute

Using TSP Data to Evaluate Your Project Performance

Shigeru Sasao
William Nichols
James McCurley

September 2010

TECHNICAL REPORT
CMU/SEI-2010-TR-038
ESC-TR-2010-103

Software Engineering Process Management
Unlimited distribution subject to the copyright.

<http://www.sei.cmu.edu>



This report was prepared for the

SEI Administrative Agent
ESC/XPB
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2010 Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. This document may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-10-C-0008 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI publications, please visit the library on the SEI website (www.sei.cmu.edu/library).

Table of Contents

Acknowledgments	vii
Abstract	ix
1 Introduction	1
2 Characteristics of a Sample of TSP Projects	3
2.1 Database Integrity	3
2.2 Team Size	4
2.3 Project Duration	5
2.4 System Size	6
2.5 Time in Phase	7
2.6 Defects	8
3 Fidelity	11
3.1 Fidelity Measures	11
3.1.1 Measures Indicating Early Defect Removal	11
3.1.2 Measures Indicating Data Quality	13
3.2 Analysis of Fidelity	16
3.2.1 Analysis of Early Defect Removal	17
3.2.2 Analysis of Data Quality	20
3.2.3 Summary of Fidelity Analysis	24
4 Performance	25
4.1 Performance Measures	25
4.2 Analysis of Performance	27
4.2.1 Analysis of Cost	27
4.2.2 Analysis of Quality	28
4.3 Analysis of Earned Product	29
4.3.1 Summary of Performance Analysis	30
5 Future Research	31
6 Conclusions	33
Appendix A Benford's Law	35
References/Bibliography	37

List of Figures

Figure 1:	Timeline of Four Related TSP Tools	3
Figure 2:	Histogram of Team Size	4
Figure 3:	Histogram of Project Duration	5
Figure 4:	Histogram of System Size in Lines of Code (LOC)	6
Figure 5:	Leading Digit Plot Showing Largest Point of Deviation	13
Figure 6:	Histogram of Planned Phases	17
Figure 7:	Histogram of Planned A/FR	18
Figure 8:	Histogram of Actual A/FR	19
Figure 9:	Histogram of Leading and Trailing Digit Deviation for Effort	20
Figure 10:	Trailing Digit Plot for a Team Collecting Data in Real Time	21
Figure 11:	Trailing Digit Plot for a Team Not Collecting Data in Real Time	22
Figure 12:	Histogram of Leading and Trailing Digit Deviation for Defects	23
Figure 13:	Trailing Digit Plot of Defect Data	23
Figure 14:	Histogram of the Cost Deviation	28
Figure 15:	Histogram of Earned Product	30
Figure 16:	Yield Before System Test to Fraction of Effort in System Test	31
Figure 17:	Actual A/FR to Yield Before System Test	32
Figure 18:	Frequency of Digits from 1 to 9 as the First Significant Digit	36

List of Tables

Table 1:	Descriptive Statistics of Team Size	4
Table 2:	Descriptive Statistics of Project Duration in Days	5
Table 3:	Descriptive Statistics of System Size in Lines of Code (LOC)	6
Table 4:	Average Percent of Time Spent in Each Phase	7
Table 5:	System Size and Total Number of Logged Defects per Team	8
Table 6:	Average Percent of Total Defects Injected per Phase Across Projects	8
Table 7:	Average Percent of Total Defects Removed per Phase Across Projects	9
Table 8:	Descriptive Statistics of Planned Phases	17
Table 9:	Descriptive Statistics of Planned A/FR	18
Table 10:	Descriptive Statistics of Actual A/FR	19
Table 11:	Descriptive Statistics of Leading and Trailing Digit Deviation for Effort	20
Table 12:	Descriptive Statistics of Leading and Trailing Digit Deviation for Defects	22
Table 13:	Descriptive Statistics of Cost Deviation	27
Table 14:	System Size, Yield Before System Test, and Fraction of Effort in System Test per Team	29
Table 15:	Descriptive Statistics for Earned Product	29
Table 16:	Probability of Observing Digits 1 to 9 as the First Significant Digit	35

Acknowledgments

The authors would first like to thank David Zubrow and Erin Harper for providing valuable feedback throughout the course of this work. Thanks also to Mark Kasunic for his expertise and suggestions for effective formatting of the report. A special thanks to David Garlan, Eduardo Miranda, Grace Lewis, and Jane Miller for making Shigeru's collaboration with the Software Engineering Institute (SEI) possible.

Abstract

The Team Software Process (TSP) provides a framework to predictably and effectively build software-intensive products. It relies on data collected by team members to provide insight into how a software project is operating. For this paper, an exploratory data analysis was conducted to investigate other ways that TSP data could be used. A set of measures was determined that allow analyses of TSP projects in terms of their fidelity to the TSP process and their project performance. These measures were applied to a data set of 41 TSP projects from an organization to identify their strengths and weaknesses. Software engineering teams already using TSP for software development can use the measures provided in this report to gain further insight into their projects.

1 Introduction

The Team Software Process (TSP) provides a framework to predictably and effectively build software-intensive products [Humphrey 2010]. TSP has shown to be particularly effective for medium to large projects where intricate coordination is required among team members with different skill sets [Jones 2009]. TSP relies heavily on data collected by team members to provide insight into how the project is operating. Team members can identify areas of the project where they are performing well, and areas where they are not, by analyzing the data. They can then tailor their processes to strengthen the areas where they are not performing well, to continuously improve their performance.

For this report, an exploratory data analysis was conducted to provide insight into the types of information an organization's TSP data can provide. A data set consisting of 41 TSP projects from one organization was analyzed for fidelity and performance using a set of derived measures from TSP data. Fidelity indicates how well the teams followed the TSP process. Project performance is determined by factors related to cost, quality, and functional completion. From the analysis, the process maturity, strengths, and weaknesses of the organization became clear.

Software engineering teams already using TSP for software development can use the measures provided in this report to gain further insight into their projects. Also, given a large data set, TSP projects can be classified into high performers and low performers using the provided measures. The measures for such high and low performers can be analyzed for purposes of performance benchmarking [Kasunic 2008].

In the following section, the basic statistics of the data set are introduced to familiarize you with the data. Then, the results of analyzing the projects for their fidelity and performance are presented.

2 Characteristics of a Sample of TSP Projects

This section provides an overview of the data set that was analyzed. Becoming familiar with these basic statistics will help you better understand the analyses of fidelity and performance provided in upcoming sections.

2.1 Database Integrity

A data set based on 41 software project data from one organization was used. All projects in the set used TSP as their software development process. The data was obtained in the form of a Microsoft Access database. The database was consolidated from individual projects contained in the TSP Tool, which is a Microsoft Excel based tool provided by the SEI for managing TSP projects.

Initially, the database contained information from 44 TSP Tools. However, two projects contained identical data in terms of team, architecture, tasks, and schedule. One of the two identical projects was discarded prior to analysis. Two other projects contained only data from the TSP launch before the plan was put into use, so they were also discarded prior to analysis.

Also, it was apparent that some projects were related after observing the names and the schedules of the projects. For example, Figure 1 shows a timeline of four TSP Tools with identical project names, distinguished by version numbers such as 3.1 and 3.2. The timeline could indicate that a separate TSP Tool was used for each of the four iterations of a single project. However, after further investigation of similar patterns in the 41 projects, it was found that none of them contained duplicate architectural components or tasks. Also, each data point contained a full development phase, from design to system testing, showing a preference by the organization to have smaller projects separated by versions of software. Each of these projects was retained and treated as a separate project. Thus, all 41 data points were treated as independent for this analysis.

Finally, as will be indicated during the analyses performed, some projects were missing certain types of data, notably time log data, defect data, and size data. Time log data was unavailable for 5 of the 41 projects. Only 15 projects collected defect data, out of which 3 teams had recorded defects found in phases that were not a part of their plan. Also, 2 teams failed to record size data.

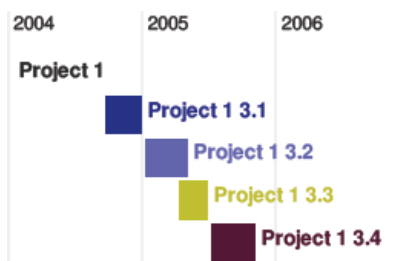


Figure 1: Timeline of Four Related TSP Tools

2.2 Team Size

The statistics for team size for the 41 projects are summarized in Table 1. The average size of the team was about 9 members, with the smallest team consisting of 4 members and the largest team consisting of 19 members. Only 12 of the 41 projects (less than 30%) had teams larger than 10, as shown in Figure 2.

Min	4.0
1 st Quartile	6.0
Median	8.0
Mean	9.3
3 rd Quartile	12.0
Max	19.0

Table 1: Descriptive Statistics of Team Size

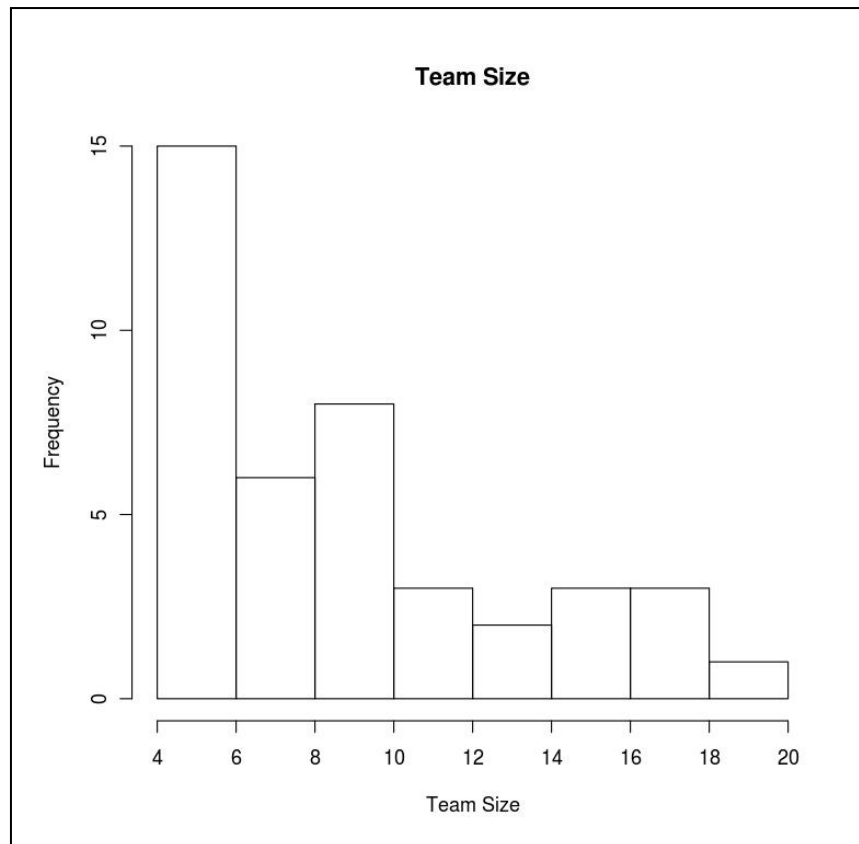


Figure 2: Histogram of Team Size

2.3 Project Duration

The statistics for project duration (in days) are summarized in Table 2. The duration of the project was defined as the number of days from the start date to the end date of the project as indicated in the TSP Tool. None of the projects had breaks between the start and end dates. The average project was about 127 days, or 4 months, with the smallest project being slightly less than a month and the largest being almost one year. As shown in the histogram in Figure 3, most of the projects were between 50 and 150 days in duration.

Min	28.0
1 st Quartile	71.8
Median	108.5
Mean	126.5
3 rd Quartile	145.2
Max	350.0

Table 2: Descriptive Statistics of Project Duration in Days

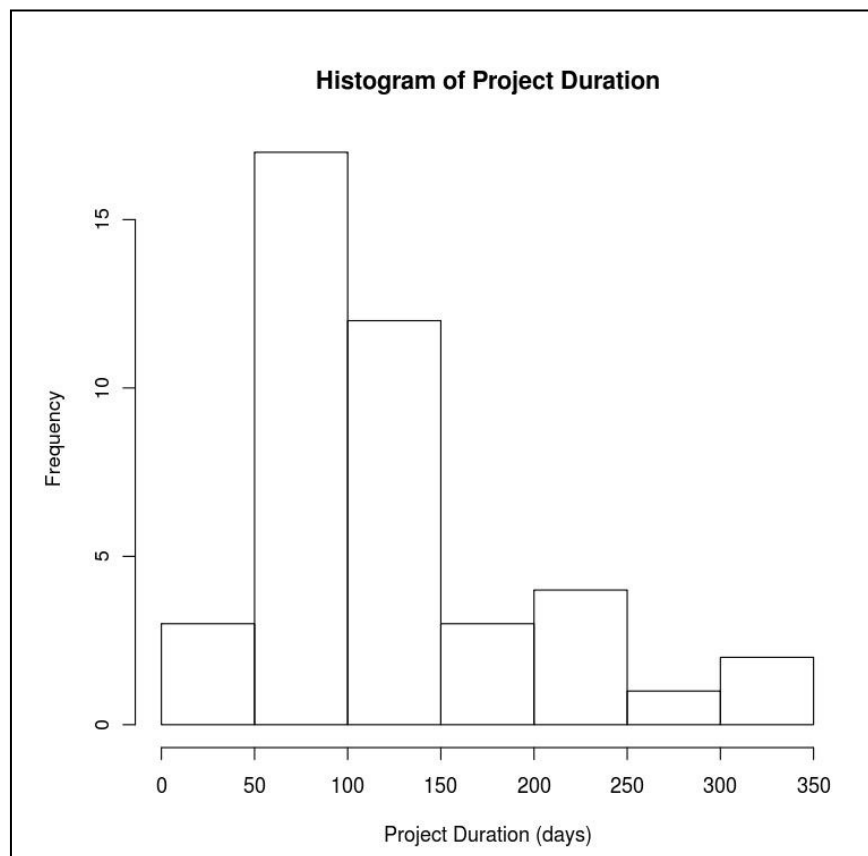


Figure 3: Histogram of Project Duration

2.4 System Size

The statistics for system size are summarized in Table 3. System size is defined as the size of the entire architecture in lines of code (LOC). From the TSP Tool, the system size was extracted as the summation of the components in the SUMS forms that used LOC as the size measure. Size data was unavailable for 2 projects, so the statistics were derived from the other 39 projects. The average system size was about 9,000 LOC or 9 Kilo LOC (KLOC), with the smallest being 265 LOC and the largest being about 60 KLOC. Figure 4 shows a histogram of system size. Thirty of the 39 projects were less than 10 KLOC, and a more detailed histogram of these systems is also shown in Figure 4.

Min	265.0
1 st Quartile	2,682.0
Median	4,340.0
Mean	9,011.0
3 rd Quartile	9,044.0
Max	60,600.0

Table 3: Descriptive Statistics of System Size in Lines of Code (LOC)

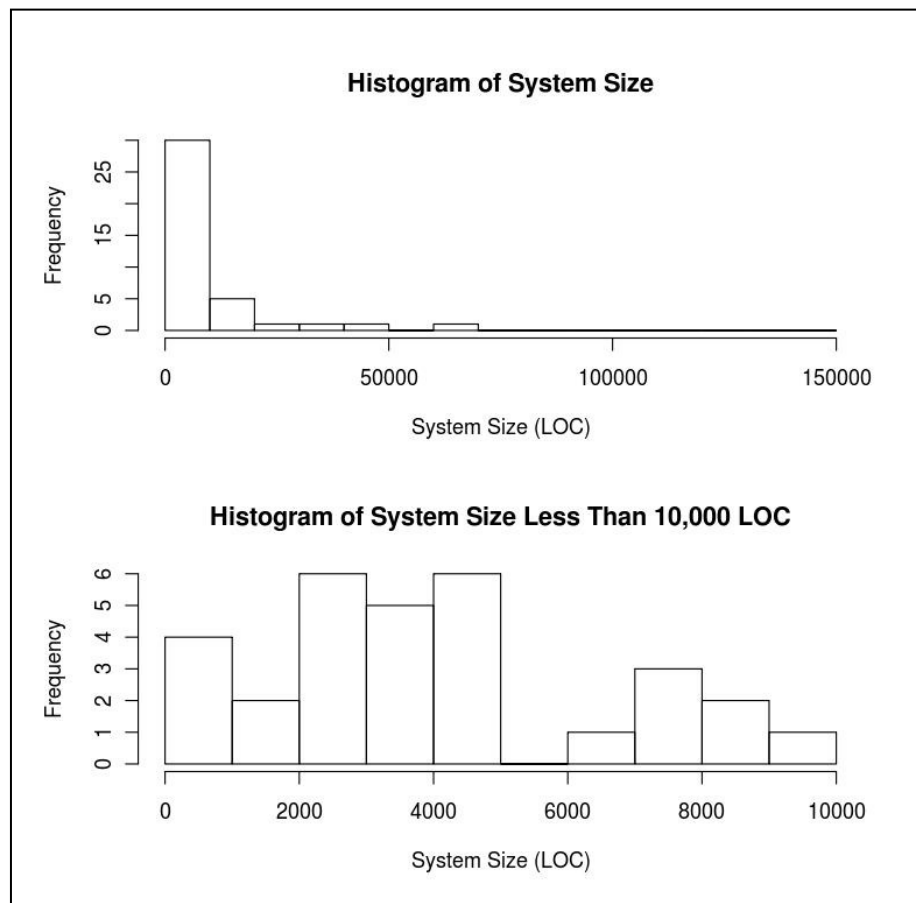


Figure 4: Histogram of System Size in Lines of Code (LOC)

2.5 Time in Phase

Table 4 shows the average percent of time spent in each development phase. The phases shown in the table are standard phases used in TSP. Testing phases, which include test development, unit test, build and integration, system testing, and acceptance testing combined for the largest portion with about 42% of the total effort. The design phases accounted for about 17% of the total effort, and coding phases accounted for about 19%. All other phases accounted for about 22% of the total effort.

Type of Phase	Phase	Time in Phase (%)
Planning	Launch and Strategy	5.91
	Planning	1.88
	System Test Plan	2.00
	Integration Test Plan	0.57
Requirements	Requirements	1.65
	Requirements Inspection	1.63
Design	High-Level Design	1.55
	High-Level Design Inspection	1.00
	Detailed Design	7.92
	Detailed Design Review	2.85
	Detailed Design Inspection	3.48
Code	Code	11.22
	Code Review	3.66
	Code Inspection	3.56
	Compile	0.91
Test	Test Development	7.91
	Unit Test	5.66
	Build and Integration Test	15.43
	System Test	12.67
	Acceptance Test	0.18
Other	Management and Miscellaneous	5.21
	Documentation	2.58
	Postmortem	0.47
	Product Life	0.08

Table 4: Average Percent of Time Spent in Each Phase

2.6 Defects

Out of the 41 projects examined, only 15 projects tracked defect data in LOGD, the form recommended by TSP. Using LOGD, every defect found in every phase of the project is recorded in the defect log. Table 5 shows the system size and total number of logged defects per team along with logged defects per KLOC. It is important to note that the logged defects per KLOC is a measure of defects found during development, not a measure of the post-production defects. The system sizes for teams 1 and 11 were unavailable. Table 6 shows the average percent of total defects injected per phase, and Table 7 shows the average percent of total defects removed per phase. Among the projects, defects were mostly injected in the coding phase and detailed design phase. On average, appraisals (reviews and inspections) were successful in capturing half of the defects.

ID	System Size (LOC)	Defects	Defects/KLOC
1	-	254	-
2	265.0	125	471.7
3	844.0	41	48.58
4	2,467.0	43	17.43
5	2,696.0	90	33.38
6	2,090.0	9	4.31
7	3,893.0	92	23.63
8	2,772.0	158	57.00
9	2,668.0	13	4.87
10	3,606.0	196	54.35
11	-	142	-
12	7,331.0	60	8.18
13	9,220.0	280	30.37
14	7,669.0	196	25.56
15	49,333.0	1,447	29.33

Table 5: System Size and Total Number of Logged Defects per Team

Phase	Defects Injected (%)
Requirements	2.23
High-Level Design	1.42
Test Development	1.65
Detailed Design	28.93
Code	62.20
Unit Test	1.98
Other	1.59

Table 6: Average Percent of Total Defects Injected per Phase Across Projects

Phase	Defects Removed (%)
Requirements Inspection	1.56
High Level Design Inspection	0.39
Detailed Design Review	12.09
Detailed Design Inspection	9.86
Code Review	22.53
Code Inspection	10.34
Unit Test	15.82
Integration Test	8.03
System Test	11.70
Other	7.68

Table 7: Average Percent of Total Defects Removed per Phase Across Projects

3 Fidelity

Fidelity indicates how faithfully the teams followed the TSP process. In this section, the measures that were used to analyze the fidelity of the TSP projects are defined. The measures are then applied to the data set introduced in the previous section.

3.1 Fidelity Measures

The first three measures indicate whether the project's development phases support early removal of defects. TSP is based on studies indicating that reviews and inspections are more efficient than testing [Humphrey 1995]. Defects removed in earlier phases of development are cheaper and lead to a higher quality product. The following are the three measures of early defect removal:

- Planned Phases
- Planned A/FR
- Actual A/FR

Planned and actual time is fundamental to these process fidelity measures. Therefore, reliability of the time data becomes a concern. TSP urges team members to collect their data in real time. Data reported from memory after the fact are usually guesses or rounded and not as accurate, leading to poor data quality. The four measures below quantify this aspect of data quality (and also this aspect of process fidelity) by examining the deviation of digits from the expected distribution, such as Benford's law. Further information about Benford's law can be found in see Appendix A. The following are the measures related to data quality:

- Leading Digit Deviation for Effort
- Trailing Digit Deviation for Effort
- Leading Digit Deviation for Defects
- Trailing Digit Deviation for Defects

All seven measures related to fidelity are explained below in detail.

3.1.1 Measures Indicating Early Defect Removal

1. *Planned Phases*

Planned phases is related to whether a team has planned for a competent software development process. The following nine phases are required:

- design
- design review
- design inspections
- code
- code review
- code inspections
- unit testing

- integration testing
- system testing

Whether or not a team has done this planning can be checked by investigating the project's planned *percent time in phase* [Humphrey 1995]. If the project has at least 1% of its overall planned effort in each phase, those phases can be marked as having been included in the plan. To convert the *planned phases* into a numerical value, the following formula is used:

$$\text{Planned Phases} = \sum_{i=1}^9 x_i$$

where x_i is the binary indicator (1 or 0) indicating whether the plan contained phase i .

2. *Planned A/FR*

$$A/FR = \frac{DLDR + DLDINSP + CR + CODEINSP}{COMPILE + UT + IT + ST}$$

where

<i>DLDR</i>	is the planned time in phase for detailed design review.
<i>DLDINSP</i>	is the planned time in phase for detailed design inspection.
<i>CR</i>	is the planned time in phase for code review.
<i>CODEINSP</i>	is the planned time in phase for code inspection.
<i>COMPILE</i>	is the planned time in phase for compile.
<i>UT</i>	is the planned time in phase for unit testing.
<i>IT</i>	is the planned time in phase for integration testing.
<i>ST</i>	is the planned time in phase for system testing.

The *planned appraisal to failure ratio (A/FR)* is defined as the ratio between the percent of total development time planned for reviews and inspections to the percent of total development time planned for compile and test [Humphrey 1995]. This measure can be obtained from the above formula using *planned time in phase* as input. If the *planned A/FR* is low, that indicates that not enough time was planned for appraisals and the software may have low quality.

3. *Actual A/FR*

The formula for the *actual A/FR* is the same as the formula for *planned A/FR* above, except for the use of *actual time in phase* instead of *planned time in phase*. While the *planned A/FR* indicates whether sufficient appraisals were planned, the *actual A/FR* indicates if the team actually conducted sufficient reviews and inspections while executing the plan.

3.1.2 Measures Indicating Data Quality

1. Leading Digit Deviation for Effort

The *leading digit deviation for effort* is a measure of the quality of the effort data that was collected by the team members in their time log. In order to assure high quality of data, PSP/TSP urges members to collect their data in real time. This measure uses Benford's law to detect whether the team members are in fact collecting data in real time and not manipulating the data after the fact.

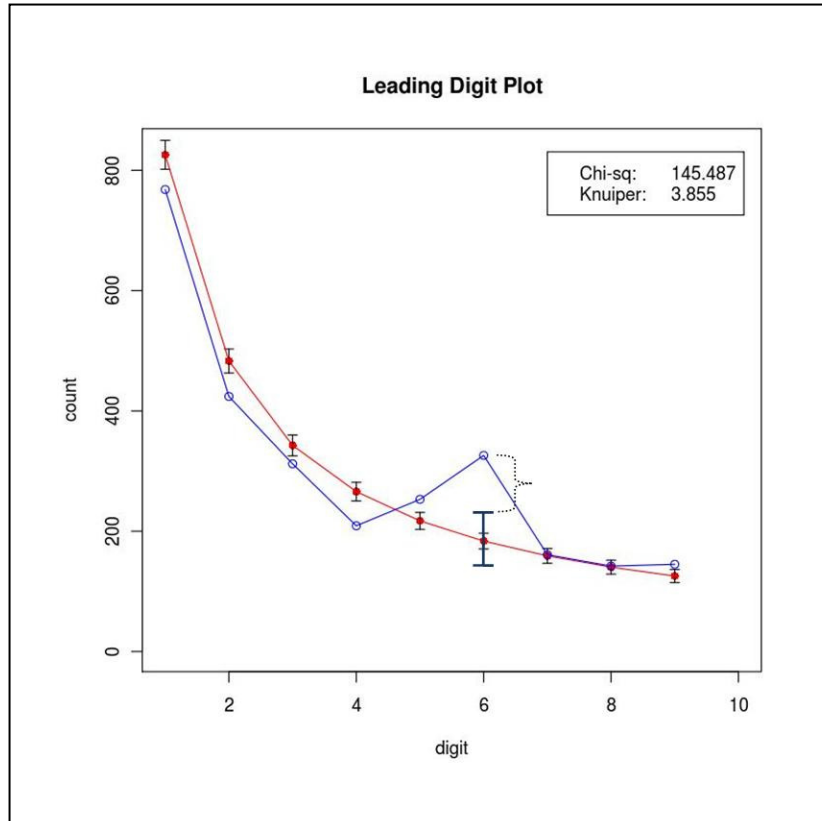


Figure 5: Leading Digit Plot Showing Largest Point of Deviation

Figure 5 shows a plot of the leading digits of the actual effort data (delta time) collected from a project's task list. The red line represents the expected frequency of digits according to the Benford's law (Appendix A). The blue line represents the observed frequency of digits, based on actual data taken from the project's time log. The black vertical intervals surrounding the expected frequencies signify one standard deviation from the expected frequency. In Figure 5, the largest deviation between the expected and the observed frequency occurs at digit 6. The *leading digit deviation for effort* is a measure related to the distance between the observed frequency and two standard deviations from the expected frequency of the largest deviation point, as indicated in the diagram by a dashed bracket.

Some vocabulary and variables need to be defined before explaining how to calculate this measure. First, let n be the *total number of observations*, which is the number of tasks in the task list. A *leading digit* is the left most digit of a number. For example, the set of numbers:

$$N = \{10, 100, 30, 25, 90, 4\}$$

has the following *leading digits*:

$$L = \{1, 1, 3, 2, 9, 4\}$$

Benford's distribution for the numbers 1 through 9 is defined as the following vector:

$$B = \begin{bmatrix} \log_{10}(1 + \frac{1}{1}) \\ \log_{10}(1 + \frac{1}{2}) \\ \vdots \\ \log_{10}(1 + \frac{1}{9}) \end{bmatrix}$$

The product $n * B$, which is the *expected frequency of digits*, is represented in Figure 5 by the points connected by the red line. Given B and n and using a binomial distribution, the standard deviation for each digit's expected frequency is:

$$S_B = \begin{bmatrix} (n * B_1 * (1 - B_1))^{\frac{1}{2}} \\ (n * B_2 * (1 - B_2))^{\frac{1}{2}} \\ \vdots \\ (n * B_9 * (1 - B_9))^{\frac{1}{2}} \end{bmatrix}$$

The standard deviation is represented in Figure 5 by the vertical black intervals. The actual *observed frequency of digits* is defined as:

$$O = \begin{bmatrix} o_1 \\ o_2 \\ \vdots \\ o_9 \end{bmatrix}$$

where o_i is the observed frequency of digit i . For example, for the above example set of *leading digits* L , o_1 is 2, since there are two occurrences of digit 1. The *observed frequency of digits* is derived from the *leading digits* of the actual effort data (delta time) collected from the project's task list. The *observed frequency of digits* is represented in Figure 5 by the points connected by the blue line.

The *absolute value function* (*abs*) is an element-wise function that takes the absolute value of each element in the vector. For example, for the following vector:

$$A = \begin{bmatrix} -2 \\ 5 \\ -1 \\ 6 \end{bmatrix}$$

applying the absolute value function, $abs(A)$, will return:

$$abs(A) = \begin{bmatrix} 2 \\ 5 \\ 1 \\ 6 \end{bmatrix}$$

Finally, the *max function* (*max*) is a function that returns the maximum value of a vector. For the above vector, $max(A)$ will return 6.

Using the above definitions, the *leading digit deviation for effort* is defined as:

$$LDDE = \frac{\max (abs(n * B - O) - 2 * S_B)}{n}$$

That is, *LDDE* is the maximum deviation in excess of two standard deviations of the expectation, divided by the number of observations. This statistic estimates the fraction of observations that were not consistent with Benford's distribution.

2. *Trailing Digit Deviation for Effort*

The *trailing digit deviation for effort* is also a measure of the quality of the effort data collected by the team members. When members do not collect data in real time as the work is being performed, the last digits are often rounded to 5 or 0. This measure will capture frequent occurrences of such rounding. Instead of Benford's law, which only applies to the leading digits, the *trailing digit deviation for effort* uses the uniform distribution to calculate the expected frequency of digits. A modification of Benford's law could be used, however, a uniform distribution is a good approximation with more than a few significant figures.

Let n be the *total number of observations*, which is the number of tasks in the task list. A *trailing digit* is the right-most digit of a number. For example, the following set of numbers:

$$N = \{32, 143, 242, 25, 92, 4\}$$

has the following *trailing digits*:

$$T = \{2, 3, 2, 5, 2, 4\}$$

A *discrete uniform distribution* with possible values $k = 1, 2, \dots, 10$ is defined as:

$$U = \begin{bmatrix} 0.1 \\ 0.1 \\ \vdots \\ 0.1 \end{bmatrix}$$

with 10 elements of value 0.1. The product $n * U$ represents the *expected frequency of digits*. Given U and n , the standard deviation for each digit's expected frequency is:

$$S_U = \begin{bmatrix} (n * U_1 * (1 - U_1))^{\frac{1}{2}} \\ (n * U_2 * (1 - U_2))^{\frac{1}{2}} \\ \vdots \\ (n * U_{10} * (1 - U_{10}))^{\frac{1}{2}} \end{bmatrix}$$

The *observed frequency of digits* is defined as:

$$O = \begin{bmatrix} o_1 \\ o_2 \\ \vdots \\ o_{10} \end{bmatrix}$$

where o_i is the observed frequency of digit i . The *observed frequency of digits* is derived from the *trailing digits* of the actual effort data (delta time) collected from the project's task

list. It is important to note that the last entry of the vector at position 10 is the occurrence of digit 0.

The *trailing digit deviation for effort* is defined as:

$$TDDE = \frac{\max (abs(n * U - O) - 2 * S_U)}{n}$$

The *TDDE* again uses the excess two standard deviations above the expectation.

3. *Leading Digit Deviation for Defects*

The *leading digit deviation for defects* uses the same formula as the *leading digit deviation for effort*, except the *observed frequency of digits* is derived from the recorded defect fix times instead of the effort time log. Thus,

$$LDDD = \frac{\max (abs(n * B - O) - 2 * S_B)}{n}$$

where *observed frequency of digits* is derived from recorded defect fix times and n is the total number of defects in the defect log. This measure is an indication of the quality of the defect fix times being collected by the team members.

4. *Trailing Digit Deviation for Defects*

The *trailing digit deviation for defects* uses the same formula as the *trailing digit deviation for effort*, except the *observed frequency of digits* is derived from the recorded defect fix times instead of the effort time log. Thus,

$$TDDD = \frac{\max (abs(n * U - O) - 2 * S_U)}{n}$$

where *observed frequency of digits* is derived from recorded defect fix times and n is the total number of defects in the defect log. This measure is also an indication of the quality of the defect fix times being collected by the team members.

To get a better intuitive numerical representation of the deviation measures, they can be converted into a scale from 0 to 100 by using the following formula:

$$Digit\ Deviation\ Score = (1 - x) * 100$$

where x is one of the four measures of deviation.

3.2 Analysis of Fidelity

The above measures for fidelity were applied to the data set introduced in Section 2, and the results are discussed below.

3.2.1 Analysis of Early Defect Removal

Table 8 shows the basic statistics of the *planned phases* measure for the 41 projects, and Figure 6 shows the histogram of the *planned phases*. As seen from Figure 6, 19 of the 41 projects contained all 9 phases in their project plan. Over 80% (34 of 41) of the projects had 7 or more of the 9 required phases. This indicates that most of the teams planned to conduct reviews and inspections as a part of their software development process.

Min	3.0
1 st Quartile	7.0
Median	8.0
Mean	7.78
3 rd Quartile	9.0
Max	9.0

Table 8: Descriptive Statistics of Planned Phases

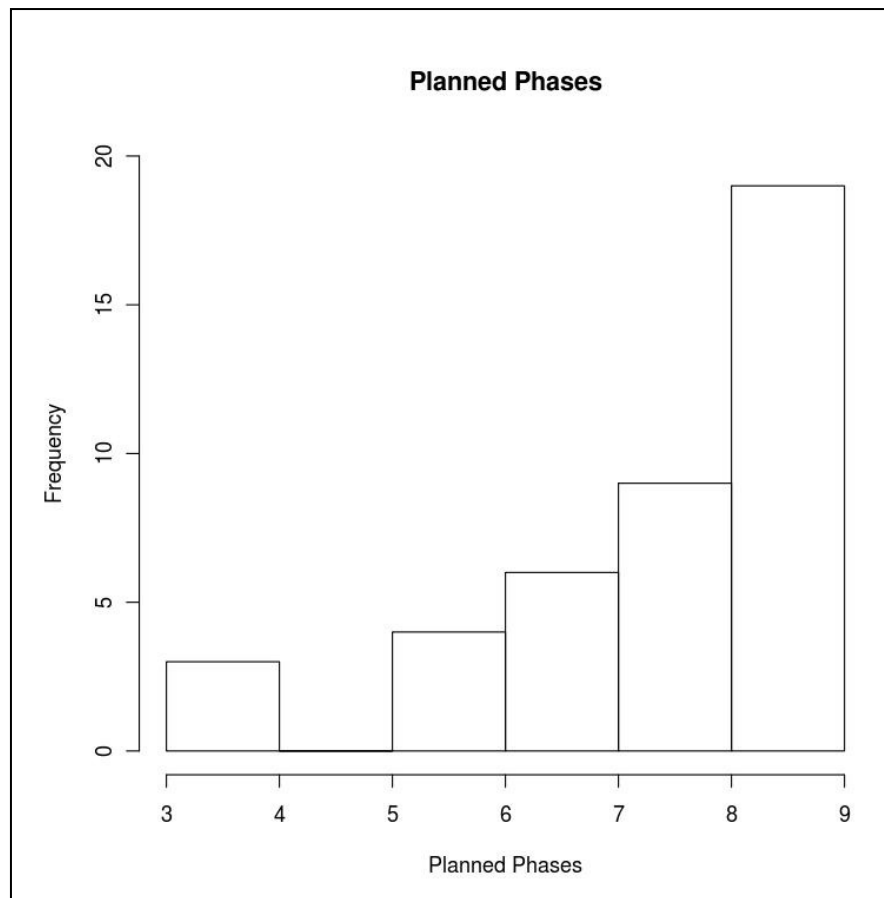


Figure 6: Histogram of Planned Phases

Table 9 and Figure 7 show the descriptive statistics and the histogram of the *planned A/FR*. Although the *planned phases* above indicate that reviews and inspections were planned as part of the project, the *planned A/FR* measure indicates that not enough time was planned for these activities. The low mean of 0.72, along with the histogram showing that most teams had a *planned A/FR* less than 1, indicates that most teams planned to spend more time in testing than appraisals.

Min	0.05
1 st Quartile	0.24
Median	0.48
Mean	0.72
3 rd Quartile	0.85
Max	3.82

Table 9: Descriptive Statistics of Planned A/FR

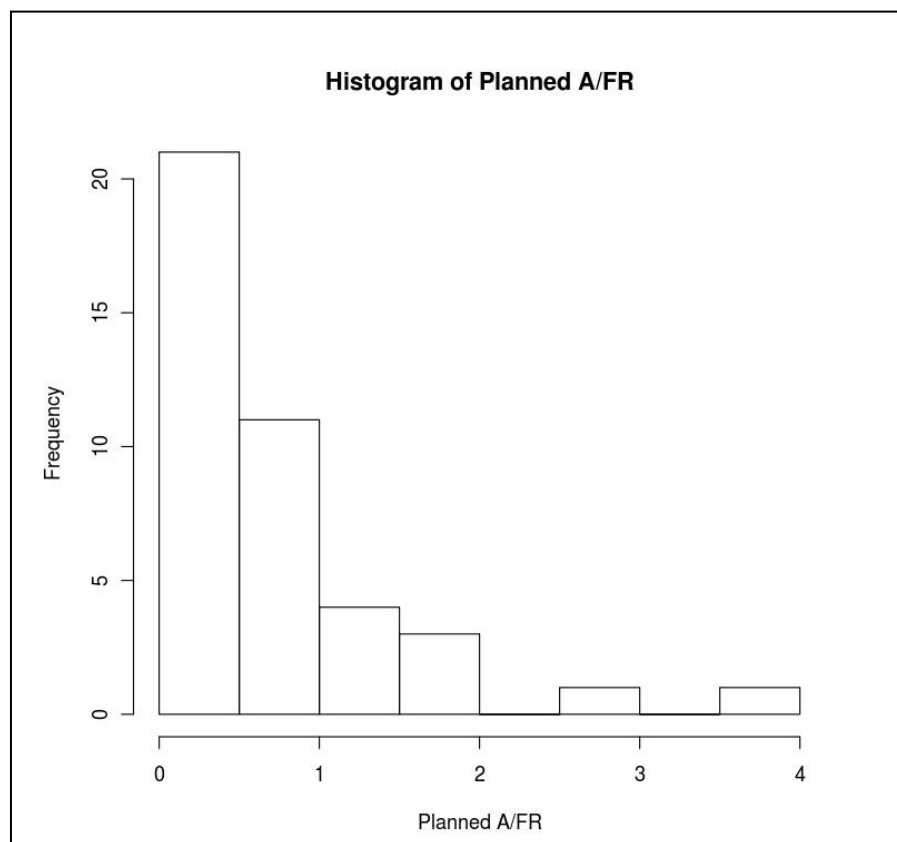


Figure 7: Histogram of Planned A/FR

Table 10 shows the descriptive statistics of the *actual A/FR* measure. The measure shows a lack of time spent in appraisals. Figure 8 shows a histogram of the *actual A/FR*, showing that most teams spent much more time testing than they did reviewing or inspecting their artifacts. It is highly probable that since the planned process did not have adequate appraisals, the teams also did not perform adequate appraisals when executing the plan.

Min	0.00
1 st Quartile	0.15
Median	0.30
Mean	0.46
3 rd Quartile	0.51
Max	4.93

Table 10: Descriptive Statistics of Actual A/FR

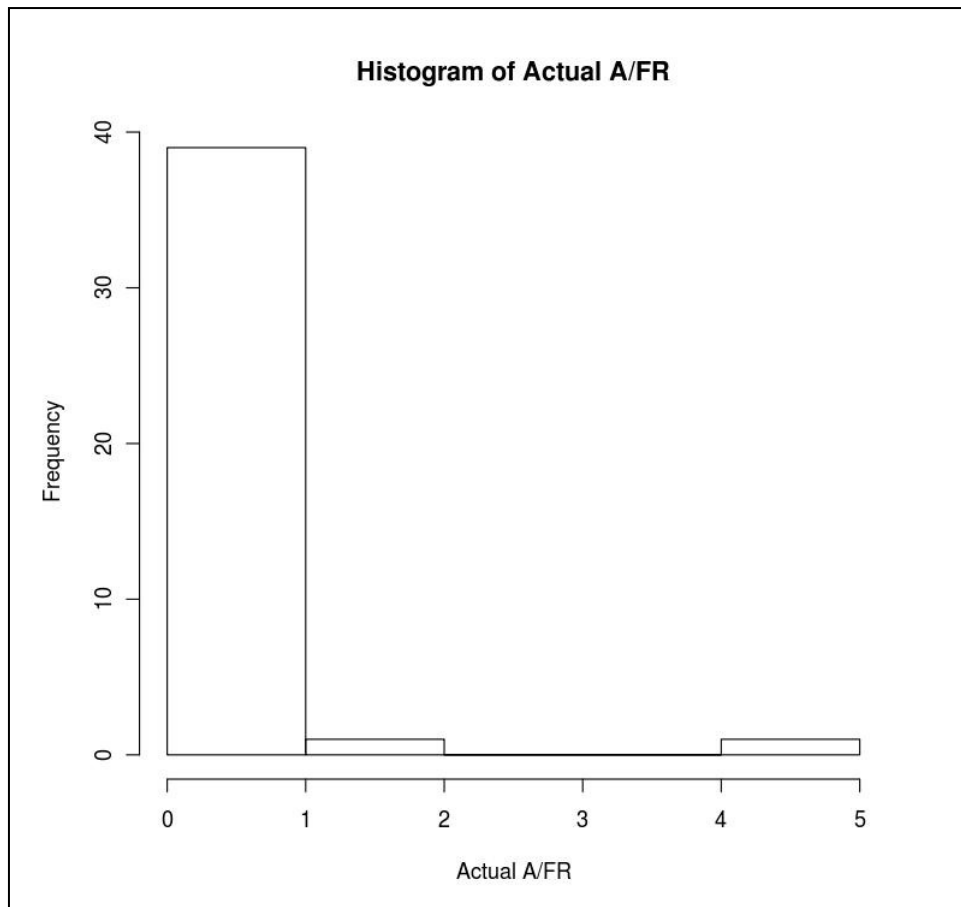


Figure 8: Histogram of Actual A/FR

3.2.2 Analysis of Data Quality

Time log information was not available for 5 of the 41 projects. Therefore, they were excluded from the analysis of the quality of the effort data. Table 11 and Figure 9 show the distribution of *leading digit deviation for effort* and *trailing digit deviation for effort*, after they have been converted to a score ranging from 0 to 100. The scores for the leading digit deviation are high, indicating that the leading digit closely followed Benford's law. However, the low mean of 79.0 for the trailing digit indicates that the recorded values were being rounded.

	Leading Digit Deviation	Trailing Digit Deviation
Min	85.3	40.7
1 st Quartile	93.2	71.9
Median	95.6	80.0
Mean	95.0	79.0
3 rd Quartile	97.2	85.4
Max	99.9	98.6

Table 11: Descriptive Statistics of Leading and Trailing Digit Deviation for Effort

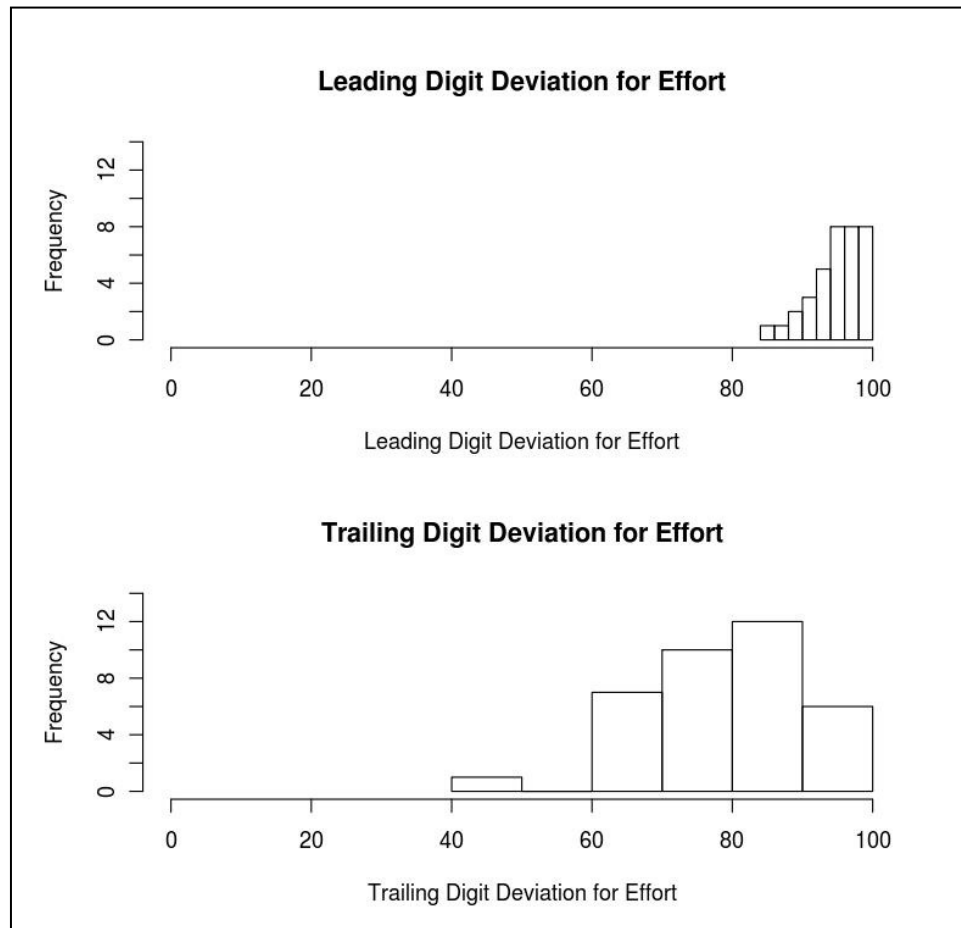


Figure 9: Histogram of Leading and Trailing Digit Deviation for Effort

Figure 10 shows an example plot of the trailing digit analysis of effort data. The red line indicates the expected distribution, while the blue line joins the points from the sample data. This figure is an example of a team that collected most of their data in real time using a stop watch timer. Although some rounding can be seen in the 0 digit, where the 0 digit is shown as the 10th digit in the plot, the blue line is not far from one standard deviation indicated by the black vertical bars. This team received a score of 98.6.

Figure 11 shows a plot of a team that did not collect data in real time. The large spikes in the blue line at 5 and 0 indicate that the team members often rounded their data at 5-minute intervals, indicating that data was not collected in real time using a timer, but was estimated. This team received a score of 80.2. The mean of 79.0 indicates that most teams rounded much of their data, and the accuracy of their data may be suspect. Rounded data, though less precise, might still be accurate. The trailing digit is, therefore, likely to produce a lower quality indicator score than the leading digit.

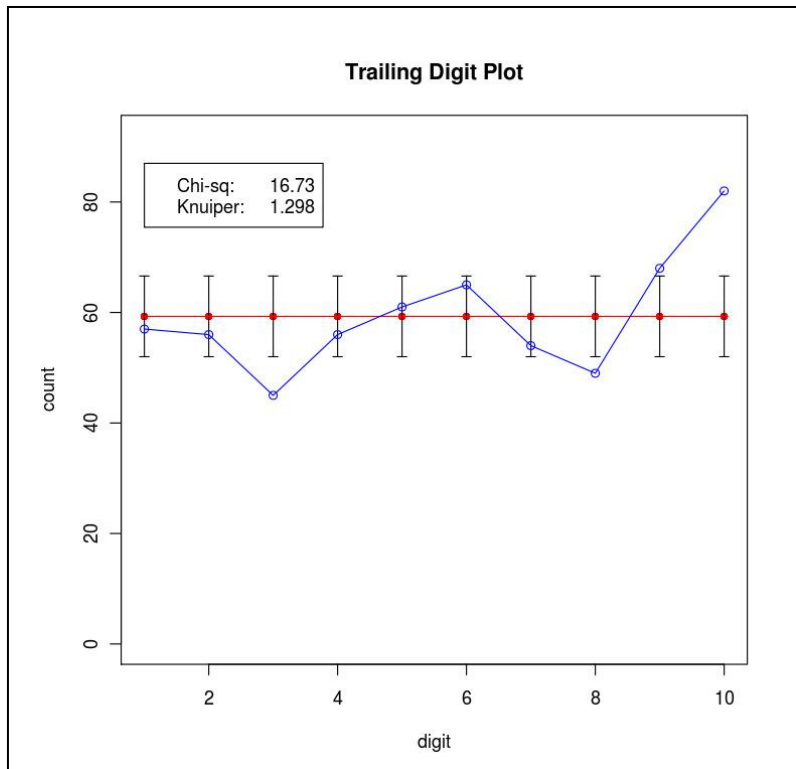


Figure 10: Trailing Digit Plot for a Team Collecting Data in Real Time

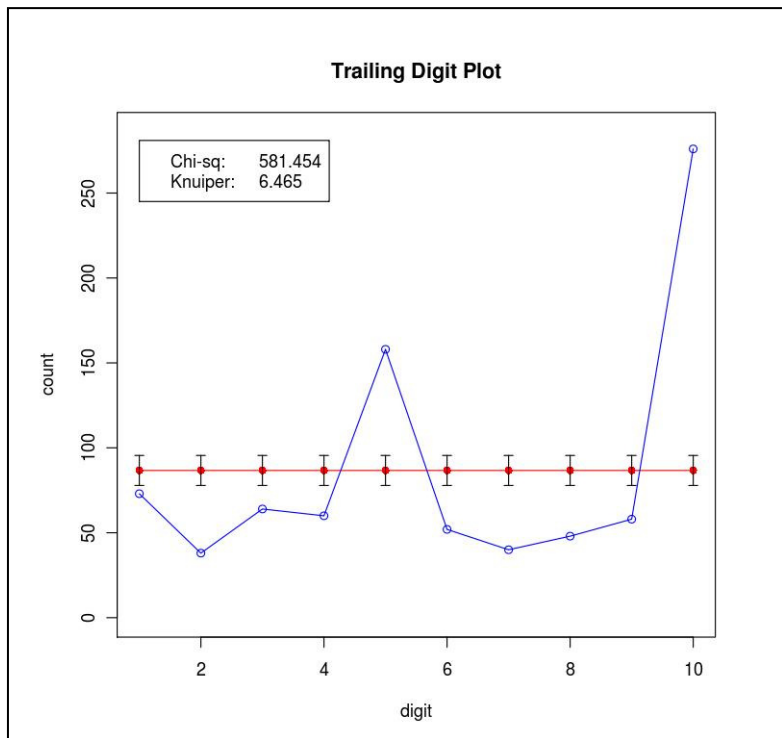


Figure 11: Trailing Digit Plot for a Team Not Collecting Data in Real Time

A somewhat higher degree of estimation was expected for defect data because a timer was not provided for collection of defect find and fix time. Table 12 and Figure 12 show the descriptive statistics and the histogram for *leading digit deviation for defects* and *trailing digit deviation for defects*. As indicated in Section 2, only 15 of the 41 projects collected defect data. Thus, the other 26 projects were discarded for this analysis. As compared to the effort data, the quality indicator of the defect data is significantly worse. For the leading digit, the mean score is 86.1, indicating that in about 14% of the sample, the leading digit deviates from the expected frequencies of the Benford distribution. This indicates that values for defect fix times were less likely to have been collected in real time. Also, the mean of 67.9 for the trailing digit deviation for defects indicates that the trailing digits were rounded. Figure 13 shows an example plot of the trailing digit, indicating severe rounding at digit 5 and 0.

	Leading Digit Deviation for Defects	Trailing Digit Deviation for Defects
Min	70.4	41.7
1 st Quartile	79.4	58.1
Median	87.9	71.0
Mean	86.1	67.9
3 rd Quartile	94.0	75.5
Max	97.9	93.0

Table 12: Descriptive Statistics of Leading and Trailing Digit Deviation for Defects

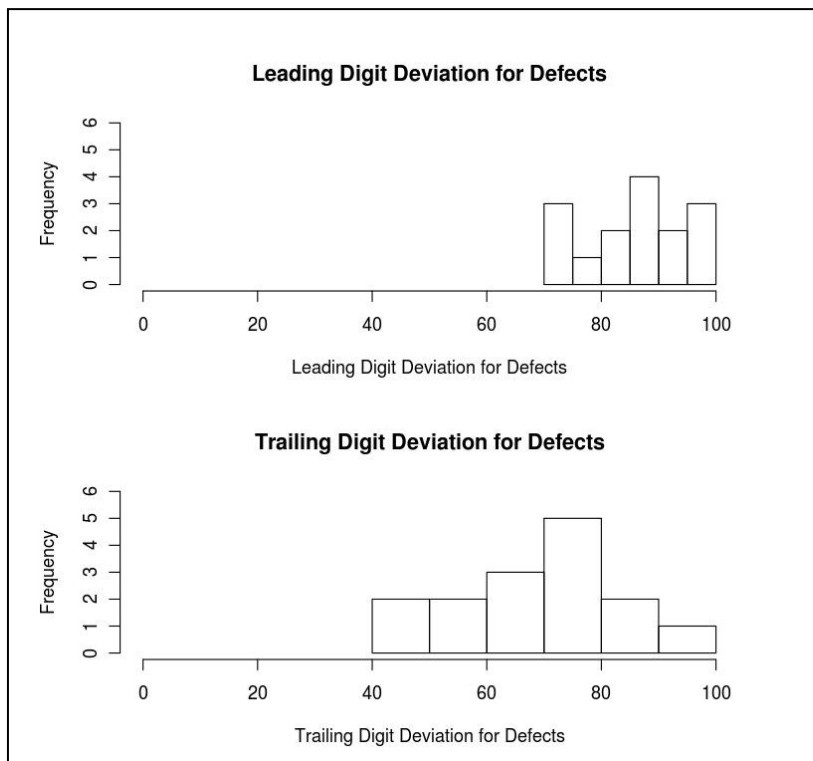


Figure 12: Histogram of Leading and Trailing Digit Deviation for Defects

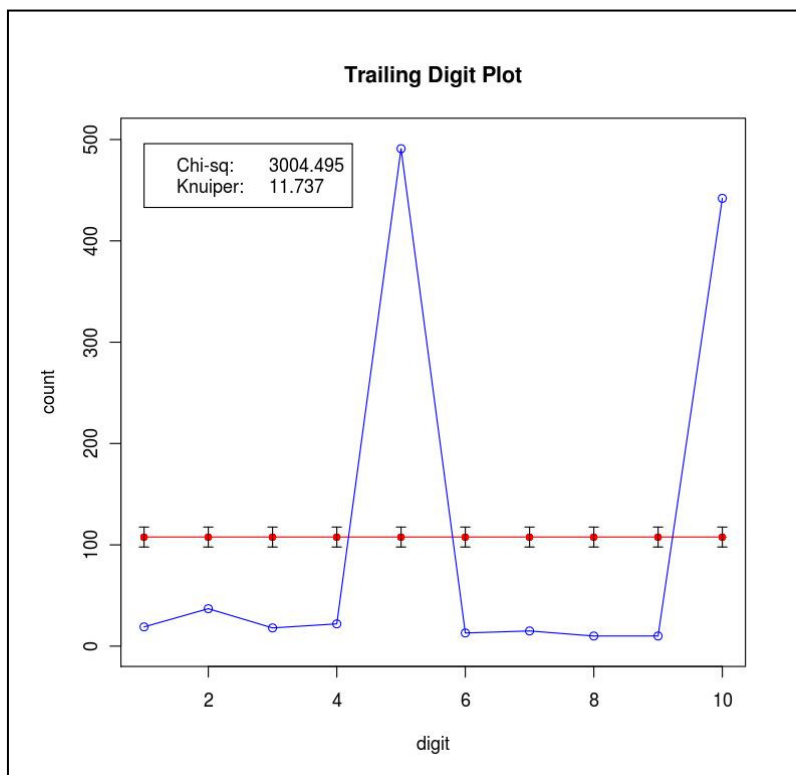


Figure 13: Trailing Digit Plot of Defect Data

3.2.3 Summary of Fidelity Analysis

The analysis of the process fidelity measures indicated that the 41 projects in the organization planned for reviews and inspections. However, they did not plan enough time for these activities, and thus did not spend enough time in appraisals as indicated by the *actual A/FR* measure. Also, the teams in the organization did not do a uniformly good job collecting their data in real time. As shown from the analysis of the trailing digit, a substantial portion of the data was estimated to rounded values of 5 and 0. For those teams that collected defect data, the quality of their defect data was not as good as the effort data. The measures for both leading and trailing digits for defect data indicated that between 15% and 30% of the data was not being collected in real time. More significantly, only 15 out of the 41 projects collected defect data. It can be inferred from the above analysis that the organization's overall process maturity for TSP was low.

4 Performance

Performance measures indicate the average performance of TSP teams. In this section, the measures that can be used to analyze the performance of TSP projects are defined. The measures are then applied to the data set introduced in Section 2.

4.1 Performance Measures

The following measures were used to analyze the performance of TSP projects:

- Cost deviation
- Yield before system test
- Fraction of effort in system test
- Earned product

The above measures indicate the project's performance regarding cost, quality, and functional completion. Each of these measures is explained in detail below.

1. Cost Deviation

$$\text{Cost Deviation} = \frac{\text{Actual Effort} - \text{Baseline Effort}}{\text{Baseline Effort}}$$

where

Actual Effort is the actual total effort for the project, in person-hours.

Baseline Effort is the planned total effort for the project, in person-hours.

This measure shows the cost performance of the TSP team. A positive value for the cost deviation shows that the project used more resources than planned to complete the project.

2. Yield Before System Test

$$YBST = 1 - \frac{ST\ Defects + Customer\ Defects}{Total\ Defects}$$

where

ST Defects is the number of defects found during the system test phase of the project.

Customer Defects is the number of defects found by the customer after release.

Total Defects is the total number of defects found for the project.

The *yield before system test* indicates the percentage of the defects found before the product is submitted for final testing. This measure is important because no system testing method can consistently find all the defects in typical software systems. An efficient and effective development process will strive to remove all defects before the beginning of final testing.

3. *Fraction of Effort in System Test*

$$\text{Fraction of ST Effort} = \frac{\text{ST Effort}}{\text{Total Effort}}$$

where

ST Effort is the hours of effort spent on the system test phase of the project.

Total Effort is the total hours of effort spent for the project.

The *fraction of effort in system test* indicates how much of the overall project effort was spent in system test. A large value for this measure typically indicates a significant amount of defect fixing and rework, indicating a low quality of the output product.

4. *Earned Product*

The *earned product* is similar to the concept of *earned value* in Earned Value Analysis (EVA) [Pressman 2005]. However, unlike *earned value*, the value is calculated based on the planned size of software components instead of the planned task effort. Software components are the individual components of the entire software architecture, represented by a row in the Program Size Summary (SUMS) Form. The *planned product* for a single software component is calculated as:

$$\text{Planned Product}_k = \frac{\text{Component Size}_k}{\text{Total Software Size}} * 100$$

where

Component Size_k is the size of a single software component *k*, usually in LOC.

Total Software Size is the size of the entire software product for the project, usually in LOC.

The *earned product* is calculated as the sum of the individual *planned products* for all completed components:

$$Earned\ Product = \sum_{i=1}^n Planned\ Product_i * Component\ Completed_i$$

where

Planned Product_i is the *planned product* of the *i*th component.

Component Completed_i is a binary indicator with value 1 if the *i*th component is completed, and 0 if it is not completed.

Components are considered completed if the actual size of the component is recorded in the form SUMS, or if all the tasks related to the component are completed. If all components that were planned to be included in the final software product are completed, the *earned product* will have a value of 100. If any of the planned components are not completed, the *earned product* will be reduced proportionally to the planned size of the unfinished component. Software development teams, when they face schedule problems, typically reduce product function. A low *earned product* is likely a result of such reduction in functionality.

4.2 Analysis of Performance

The above measures for performance were applied to the data set introduced in Section 2.

4.2.1 Analysis of Cost

The descriptive statistics of the *cost deviation* for the data set is shown in Table 13, and the histogram is shown in Figure 14. The data shows that more than half (24 of 41) of the projects completed within budget, and for those that overran their budget, they only did so by a slight margin. The most significant budget overrun was for a project that planned for 3,528 person-hours but actually used 4,909 person-hours of effort.

Min	-0.53
1 st Quartile	-0.17
Median	-0.03
Mean	-0.07
3 rd Quartile	0.07
Max	0.29

Table 13: Descriptive Statistics of Cost Deviation

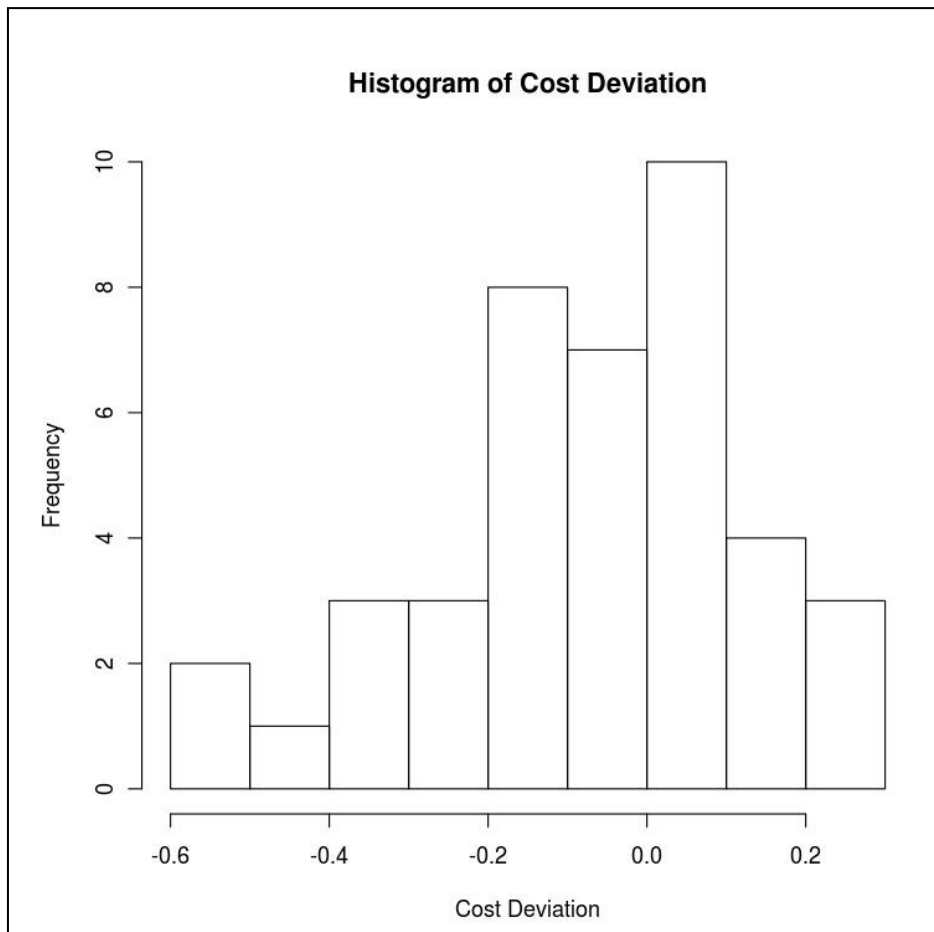


Figure 14: Histogram of the Cost Deviation

4.2.2 Analysis of Quality

As indicated in Section 2, only 15 out of the 41 projects collected defect data. Therefore, 26 projects were excluded from the analysis. Table 14 shows the system size, *yield before system test*, and *fraction of effort in system test* for each team. Seven of the 15 teams had a *fraction of effort in system test* of 0. This means that there were no tasks allocated to system test. However, out of the 7 teams that did not conduct system test, 3 teams had a *yield before system test* of less than 1. This means that even though the 3 teams captured defects in system test, there were no tasks allocated to system test. This is an issue in planning or data quality, where teams were spending time in a phase that was not a part of their plan.

For the remaining projects, Table 14 indicates that projects spent a large portion of their time in system testing. Two of the projects spent nearly half of their effort on system test, indicating that there was a significant amount of time spent on fixing defects. Project 7 in particular had a low yield of 42%, indicating that many defects were still remaining in the system when they reached system test, and 48% of the project effort was spent in system test. Projects 10 and 11 show healthy measures, with a yield of 98% and only a small portion of their spent time in system testing. However, Project 5 had a high yield of 98%, yet still spent 30% of their time in system testing.

ID	System Size (LOC)	Yield Before System Test	Fraction of Effort in System Test
1	-	1.00	0.00
2	265.0	1.00	0.00
3	844.0	0.95	0.00
4	2,467.0	1.00	0.00
5	2,696.0	0.98	0.30
6	2,090.0	1.00	0.00
7	3,893.0	0.42	0.48
8	2,772.0	0.89	0.40
9	2,668.0	0.54	0.00
10	3,606.0	0.98	0.16
11	-	0.98	0.11
12	7,331.0	0.72	0.28
13	9,220.0	0.96	0.00
14	7,669.0	0.87	0.21
15	49,333.0	0.96	0.11

Table 14: System Size, Yield Before System Test, and Fraction of Effort in System Test per Team

4.3 Analysis of Earned Product

The descriptive statistics of the *earned product* for the data set are shown in Table 15, and the histogram is shown in Figure 15.

Min	0.00
1 st Quartile	94.31
Median	100.0
Mean	91.68
3 rd Quartile	100.00
Max	100.00

Table 15: Descriptive Statistics for Earned Product

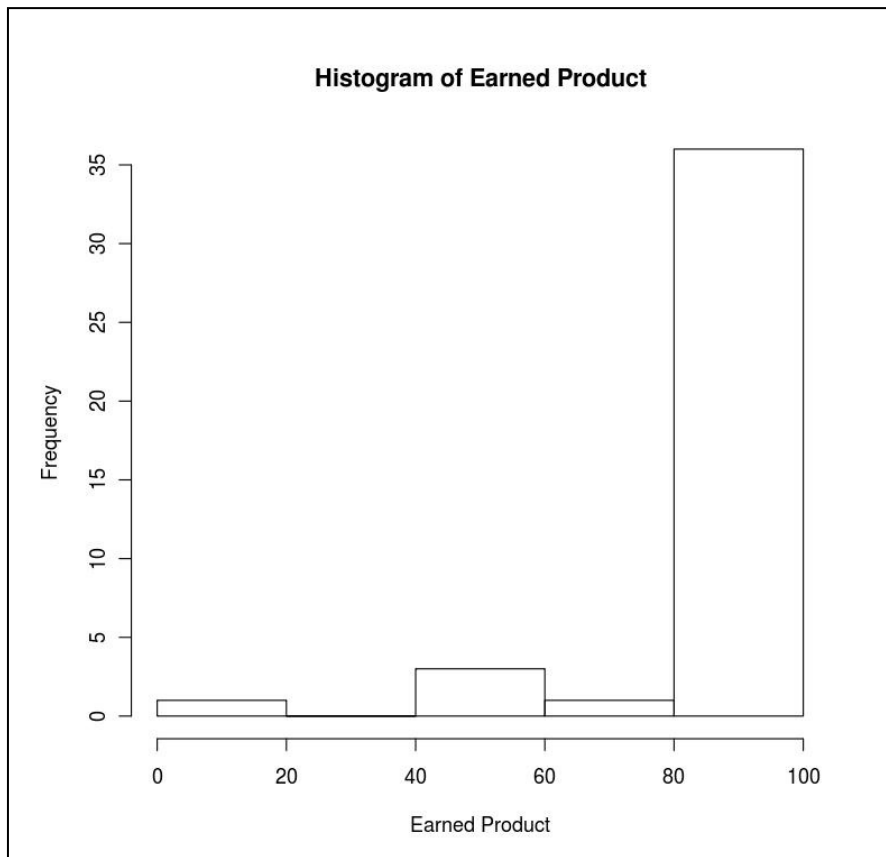


Figure 15: Histogram of Earned Product

One project had an *earned product* measure of 0. That was because they did not complete the phases that were planned for all of the components they intended to produce. Excluding this outlier, teams generally completed all or most of the components that they planned to produce. This indicates that during the project, teams did not cut scope and reduce the functionality of the system.

4.3.1 Summary of Performance Analysis

The projects in the organization were generally completed within budget without reducing the functionality of the system. However, the quality profiles of the projects are largely unknown because most teams did not collect defect data. For the defect data that were collected, there were further data quality issues related to teams spending time in phases that were not a part of their plan. The *fraction of effort in system test* shows that those teams that properly collected defect data spent a significant amount of time in system testing. This is in agreement with the observation from Section 3 that the teams generally relied on testing rather than reviews and inspections.

5 Future Research

As shown in the case study, rigor and completeness in real-time data collection are essential to obtain accurate information. When data is missing, such as the defect data in this case, it becomes difficult to obtain useful information from cross-analysis. Process maturity in TSP teams is essential to collect a complete set of data. Also, further research should be conducted into the analysis of data quality, such as the Benford analysis presented in this report.

The case study in this report was conducted on a data set of 41 projects from a single organization. It would be valuable to increase the scope of the analysis to include multiple organizations from the software industry, so we could develop a general profile of the measures and scores among industry teams using TSP. Thus, a database of TSP data containing a high volume of various industry projects would be valuable for software engineering research. Further, a best- and worst-in-class analysis of projects for teams with the highest and lowest measures of performance (cost, quality, functional completion) would be useful for obtaining performance benchmarks for TSP projects.

Finally, an analysis of yield, A/FR, and cost was conducted on 10 of the 41 projects that contained a full set of data.¹ Figure 16 shows a plot of *yield before system test*, and the *fraction of effort in system test*. Pearson's linear correlation was -0.74, and Spearman's rank correlation was -0.83. The strong correlation may indicate that if defects are removed effectively prior to system testing, teams spend a smaller percentage of their project time in system testing.

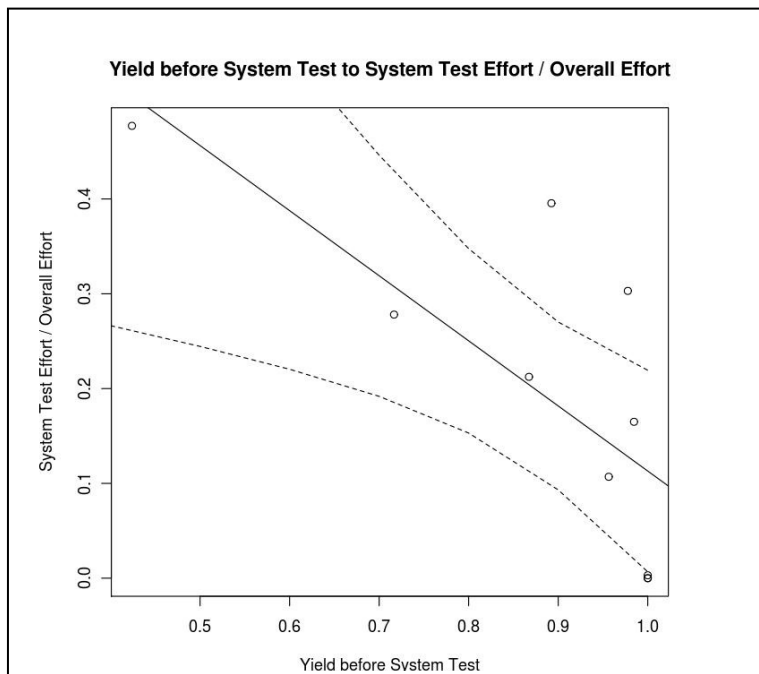


Figure 16: Yield Before System Test to Fraction of Effort in System Test

¹ Defect data was collected by 15 of the 41 projects. Out of the 15 projects examined, 2 of the projects did not record size data. From the 13 projects remaining, 3 projects recorded defects from system test, but had no task defined for system test in their task list. So, the 3 projects were discarded, and a total of 10 projects were included for analysis.

Figure 17 shows a plot between *actual A/FR* and *yield before system test*. As seen from the figure, the relationship was not linear. Pearson's linear correlation was low at 0.47, but Spearman's rank correlation was high at 0.72. This may indicate that as teams spend more time in appraisals, yield will increase. This increase will not be linear because the remaining defects become more difficult to find.

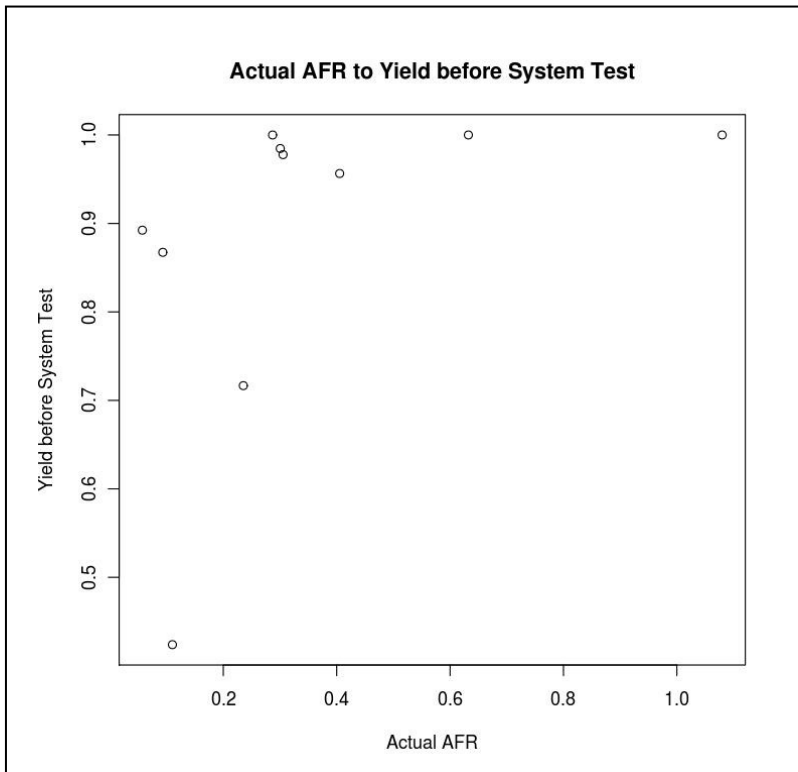


Figure 17: Actual A/FR to Yield Before System Test

The results presented here are based on only 10 data points. Again, having a rich set of complete and accurate data points would be beneficial to confirm whether such patterns are in fact general trends for software teams using TSP.

The fine-grained nature of TSP data lend themselves to analyses of the detailed activities of software projects in an unparalleled manner. Future analyses of TSP data may produce other empirically derived heuristics useful for the planning and management of software projects.

6 Conclusions

A set of measures were introduced to analyze TSP projects in terms of their fidelity and performance. The measures were applied to a data set of TSP projects from a single organization to gain insight into their project profile. In addition to the basic statistics of the data set, the weaknesses and strengths of the organization were explained through the distribution of the fidelity and performance measures among the projects. Further analysis should be conducted on an expanded set of data containing many data points from industry projects. This will enable software engineers to benchmark their performance against other projects using TSP.

Appendix A Benford's Law

Benford's law, also called the first digit phenomenon, is a law stating that certain digits occur more often than other digits in a particular position in numbers [Hill 1998, Durtschi 2004]. It was first discovered by an astronomer and mathematician, Simon Newcomb, when he noticed that library copies of logarithm tables were more worn-out in the beginning pages containing low digits [Newcomb 1881]. He concluded that the first significant digits are distributed with probabilities according to the following formula:

$$p_i = \log_{10}\left(1 + \frac{1}{d_i}\right)$$

where p_i is the probability of observing a first significant digit d_i . Using the above equation, the probability of observing digits 1-9 as the first significant digit is summarized in Table 16.

First Sig. Digit (d_i)	Probability (p_i)
1	30.1%
2	17.6%
3	12.5%
4	9.7%
5	7.9%
6	6.7%
7	5.8%
8	5.1%
9	4.6%

Table 16: Probability of Observing Digits 1 to 9 as the First Significant Digit

Later, a physicist named Frank Benford rediscovered the law independently of Newcomb and popularized it. He conducted a study of naturally occurring numbers ranging from the area of rivers to death rates and showed that the first digits are distributed closely to the Benford's law [Benford 1938].

As an example, market data was extracted from a webpage [FT 2010]. For the set of extracted numbers,

$$S = \{128.36, 126.19, 20.42, 175.51, 33.01, \dots\}$$

the first significant digit, which is the left-most digit of the number, was taken from each number in the set:

$$F = \{1, 1, 2, 1, 3, \dots\}$$

The webpage contained 74 data points, and the frequency of the first significant digit is shown in Figure 18.

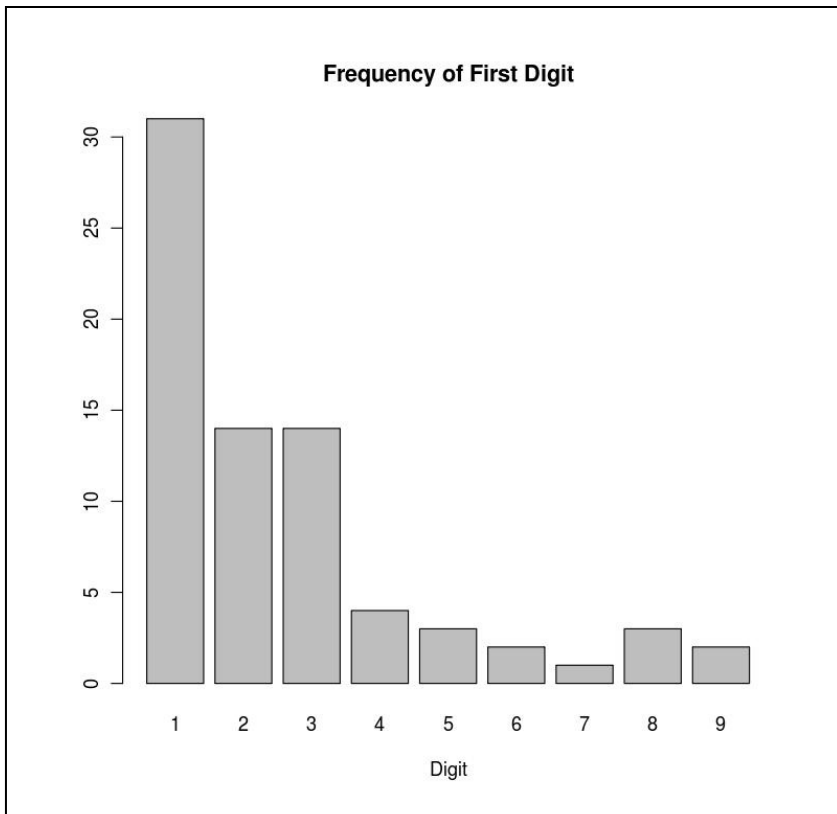


Figure 18: Frequency of Digits from 1 to 9 as the First Significant Digit

As observed, the frequency of occurrence for the digit 1 is much higher than the others. Since there were only 74 data points, the distribution does not follow Benford's law perfectly, but it is still evident that the lower digits occur more frequently than the higher digits. The following excerpt presents an intuitive explanation of Benford's law [Durtschi 2004]:

Consider the market value of a firm. If it is \$1,000,000, it will have to double in size before the first digit is a "2," in other words it needs to grow 100 percent. For the first digit to be a "3," it only needs to grow 50 percent. To be a "4" the firm must only grow 33 percent and so on. Therefore, in many distributions of financial data, which measure the size of anything from a purchase order to stock market returns, the first digit one is much further from two than eight is to nine.

Benford's law is used in forensic accounting to detect fraud [Durtschi 2004]. Most accounting data conform to Benford's law. However, in cases where a person assigns numbers, the first digits tend to be distributed uniformly rather than logarithmically. Therefore, a significant deviation from Benford's law can be seen as a potential fraud, where the numbers may have been made up or artificially modified.

References/Bibliography

[Benford 1938]

Benford, Frank. "The Law of Anomalous Numbers," *Proceedings of the American Philosophical Society*, vol. 78, no. 4 (Mar. 31, 1938), pp. 551-572, American Philosophical Society, 1938.

[Durtschi 2004]

Durtschi, Cindy; Hillison, William; & Pacini, Carl "The Effective Use of Benford's Law to Assist in Detecting Fraud in Accounting Data," *Journal of Forensic Accounting* 1524-5586/vol. V (2004), pp 17-34, R.T. Edwards, Inc., 2004.

[Fahmi 2008]

Fahmi, Syed Ahsan and Choi, Ho-Jin. "A Survey on Team Software Process Supporting Tools," ICCIT, vol. 1, pp.987-990, 2008, Third International Conference on Convergence and Hybrid Information Technology, 2008.

[FT 2010]

The Financial Times Ltd, 2010.<http://markets.ft.com/markets/overview.asp>.

[Hill 1998]

Hill, Theodore P., "The First Digit Phenomenon," *American Scientist* vol. 86, no. 4 (July-August 1998), pp 358, 1998.

[Humphrey 1995]

Humphrey, Watts. *A Discipline for Software Engineering*, Addison-Wesley, 1995.

[Humphrey 2010]

Humphrey, Watts S.; Chick, Timothy A.; Nichols, William; & Pomeroy-Huff, Marsha, *Team Software Process (TSP) Body of Knowledge (BOK)* (CMU/SEI-2010-TR-020). Carnegie Mellon University, 2010. www.sei.cmu.edu/library/abstracts/reports/10tr020.cfm

[Jones 2009]

Jones, Capers. *Software Engineering Best Practices*. McGraw-Hill Osborne Media, 2009.

[Kasunic 2008]

Kasunic, Mark. *A Data Specification for Software Project Performance Measures: Results of a Collaboration on Performance Measurement* (CMU/SEI-2008-TR-012). Carnegie Mellon University Software Engineering Institute, 2008.
www.sei.cmu.edu/library/abstracts/reports/08tr012.cfm

[Newcomb 1881]

Newcomb, Simon. "Note on the Frequency of Use of the Different Digits in Natural Numbers," *American Journal of Mathematics*, vol. 4, no. 1 (1881), pp. 39-40, The Johns Hopkins University Press, 1881.

[Pressman 2005]

Pressman, Roger S. *Software Engineering: A Practitioner's Approach*, R.S. Pressman and Associates, 2005.

[Tuma 2010]

Tuma Solutions LLC, 2010. <http://www.processdash.com/>

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE September 2010		3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Using TSP Data to Evaluate Your Project Performance			5. FUNDING NUMBERS FA8721-10-C-0008	
6. AUTHOR(S) Shigeru Sasao, William Nichols, James McCurley				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213			8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2010-TR-038	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116			10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2010-103	
11. SUPPLEMENTARY NOTES				
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS			12B DISTRIBUTION CODE	
13. ABSTRACT (MAXIMUM 200 WORDS) The Team Software Process (TSP) provides a framework to predictably and effectively build software-intensive products. It relies on data collected by team members to provide insight into how a software project is operating. For this paper, an exploratory data analysis was conducted to investigate other ways that TSP data could be used. A set of measures was determined that allow analyses of TSP projects in terms of their fidelity to the TSP process and their project performance. These measures were applied to a data set of 41 TSP projects from an organization to identify their strengths and weaknesses. Software engineering teams already using TSP for software development can use the measures provided in this report to gain further insight into their projects.				
14. SUBJECT TERMS team software process, personal software process, data, performance			15. NUMBER OF PAGES 51	
16. PRICE CODE				
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	